Key Wrapping with a Fixed Permutation

Dmitry Khovratovich

University of Luxembourg

Abstract. We present an efficient key wrapping scheme that uses a single public permutation as the basic element. As the scheme does not rely on block ciphers, it can be used on a resource-constrained device where such a permutation comes from an implemented hash function, regular (SHA-3/Keccak) or lightweight one (Quark, Photon). The scheme is capable of wrapping keys up to 1400 bits long and processing arbitrarily long headers. Our scheme easily delivers the security level of 128 bits or higher with the master key of the same length.
We use the security notion from the concept of Deterministic Authenticated Encryption (DAE) introduced by Rogaway and Shrimpton. Though the permutation is inevitably modeled as a random permutation, the resulting proof of security is short and easy to verify and hence provide a reasonable alternative to authentication modes based on block ciphers.

Keywords: Key wrapping, DAE, sponge, Keccak.

1 Introduction

Key wrapping schemes address the problem of key management in distributed systems. Security architects often limit the lifespan of keys in order to reduce the risk of the key compromise and lessen the amount of data encrypted on a single key. Hence keys are regularly updated, and an update protocol using an insecure channel must be carefully designed. Ideally, it should be simple and efficient. Practical constraints also limit, if not forbid the use of additional mechanisms such as nonce or random number generation.

Since the early years of digital cryptography, new keys are encrypted on (*wrapped with*) a long-term (*master*) key shared between a sender and a receiver. Confidentiality of the new key must be ensured and its integrity must be protected. A key might be bounded to a header, which is not encrypted (e.g., for routing purposes) but authenticated. Therefore, the key wrapping scheme is a special case of authenticated encryption with associated data (AEAD) schemes [23], where nonces and random numbers are avoided. Such a scheme may serve not only for key update, but also for a robust and misuse-resistant general purpose encryption [25].

Traditional AEAD schemes provide confidentiality (e.g., ciphertexts are indistinguishable from random strings) and data authenticity (ciphertexts constructed by an adversary must decrypt to invalid). They employ randomness or nonces and can be almost as efficient as regular encryption modes [20]. It has been clear that a deterministic scheme would require at least two passes over data to make each output bit depend on each input bit.

When the NIST addressed the key wrapping schemes in the series of recommendations (since at least 2001), its designs, later called AES-KW and AKW [21], were highly inefficient. Moreover, those schemes carried no formal security claims or proofs. This was natural, as the first formal treatment of this problem appeared only in 2006 [25] as the concept of the Deterministic Authenticated Encryption (DAE). Still, only a few key wrapping schemes have been proposed so far in DAE or similar frameworks: SIV [25], HBS [19], BTM [18], Hash-then-Encrypt [15, 22].

None of these proposals are universal solutions. AES-KW requires 12-fold as many operations as to encrypt the same amount of data, SIV needs two keys and is not parallelizable, and the Hash-then-Encrypt template can not be scaled to a general-purpose encryption mode. Third-party analysis of these constructions is often difficult because of lengthy and complicated proofs of security. The recently found flaw in the security proof of GCM [17] emphasizes the need for clarity and extensive third-party verification of provably secure schemes.

All these designs employ a block cipher, and the natural choice of AES limits their security level to 64 bits[1]. To obtain the 128-bit security or higher, one would need a block cipher with a 256-bit block or larger. Except for Threefish, a component of the Skein hash function [14], no other 256-bit blockcipher enjoyed significant attention from cryptanalysts.

However, block ciphers are not the only source of good permutations. Quite recently, the hash function Keccak [5], which employs a 1600-bit permutation, has been selected as the new standard SHA-3. We expect that it will be widely deployed in the near future, and hence its building block will be readily accessible to other cryptographic applications. On resource-constrained devices, where the space is limited, it would be very tempting to use a single cryptographic primitive, such as the Keccak permutation, for many purposes. Whenever Keccak or AES are considered too expensive for a device, some lightweight hash functions like Spongent [9] and Quark [2] are also based on a single permutation and may offer it for other schemes.

This idea also fits the recent paradigm of the *permutation-based cryptography* [11] as opposed to the blockcipher-based cryptography. From the practical point of view, it would allow to have a single permutation for all purposes, whereas it would simplify the analysis as a target for a cryptanalyst would be much simpler. Permutation-based modes of operation draw attention after the selection of Keccak, as indicated by two recent proposals for the authenticated encryption: APE(X) [8] and PPAE [7].

If the Keccak permutation is selected, the available 1600 bits are often sufficient to carry the master key, the new key, and the associated data. Hence for the design of a key wrapping scheme we could restrict the use of a permutation to a

[1] Here the key size of AES, which can be 128, 192, or 256, does not play a significant role: many modes of operation can not be proven secure as long as inputs to the blockcipher start colliding.

single call and obtain a scheme with a reasonably short proof of security. The security model, however, would be different from the one used in blockcipher-based schemes. Since we use a single permutation, the most natural is to model it as randomly drawn and hence prove the security in the random-oracle model. An alternative approach, the Even-Mansour construction, would provide the security proof in the standard model but with weaker bounds (see the further text). Though the random-permutation model is clearly more demanding to the primitive we use, we argue that a shorter and simpler security proof and increased security level would compensate the weakening of the model.

Our proposal. We present a new key wrapping scheme with a variable security level and a proof of security that is easy to verify. We call it KWF, as it is based not on a block cipher but on a fixed permutation such as those used in sponge hash functions (SHA-3/Keccak, Quark).

A wide permutation (up to 1600 bits in Keccak) easily delivers the security level of 128 bits or higher when using the key of the same length. Associated data (header) is processed with an unkeyed cryptographic hash function, possibly the same from which the permutation comes. Apart from the header processing, the scheme has no overhead over a single permutation call. We limit the message length to at max 1411 bits, but this must be sufficient for wrapping all symmetric keys and many asymmetric private keys (e.g., elliptic curve keys).

Our scheme is about as efficient as the hash function from which the permutation comes. If the associated data H is processed with the same hash function, wrapping H and M takes roughly the same time as hashing $H||M$. We recommend using a narrow-block permutation for shorter inputs. We also note that the key length can be freely chosen.

Our scheme, as well as other DAE-conformant designs, is also fine for general-purpose encryption and authenticated encryption of short inputs (though we avoid explicitly offering the scheme for general use because of length constraints). To emphasize this opportunity, we use the notion *message* for keys and other inputs that are encrypted by our scheme.

Security proof and random oracles. We accompany our design with a proof of security in the DAE framework, where we additionally allow the an adversary to query the randomly drawn permutation. Here we follow the strategy of proving the indistinguishability of the generalized Even-Mansour scheme from the random permutation [10]. Our assumption establishes the security of our scheme as long as the permutation we eventually fix has no untrivial properties (which so far holds for Keccak and other sponge functions). We tried to make our proof as simple as possible to encourage its third-party verification.

We note that there are two possible approaches to constructing and proving the security of symmetric schemes:

- Use a blockcipher as a primitive and prove the security assuming that it is a secure PRP;
- Use a concrete hash function or a public permutation as a primitive and prove the security assuming it is randomly drawn.

It is customary to consider the former approach more reliable as it is less demanding to the primitive and hence withstands a larger set of attacks against the primitive. Hence the scheme secure in the standard model is considered better than the one assuming a random oracle. However, the latter approach is arguably better from designer's point of view. One may go further and argue that, it should be easier to construct a single "good" permutation and use it, e.g., for a hash function, than to construct a family of them for a blockcipher, where a key selects a particular permutation, and all these permutations should be significantly different.

A part of our research is to investigate whether a single permutation gives an elegant scheme with a short security proof in the area of authenticated encryption. If we succeed, this would benefit the permutation-based cryptography and eventually the cryptographic community by giving various schemes with verifiable proofs.

Related work. We already mentioned other designs that aim for key wrapping and deterministic authenticated encryption. NIST has published its first key wrapping scheme around 2001 (see description in [21]). AES-KW is a sort of generalized Feistel scheme, where the key to and the header are divided into 64-bit blocks, and the round function is the AES applied to two leftmost blocks. No analysis of this scheme has been published, though it is believed [25] that the security level is somewhere between 64 and 128 bits. There are several modifications to this scheme, known as KW and KWP, and a special version that uses Triple-DES.

Rogaway and Shrimpton proposed the scheme SIV, which computes MAC of the message and header with a PRF under key K_1 and uses it as the IV in an IV-based encryption scheme with key K_2 [25]. The concrete proposal invokes CMAC and CTR. The scheme SIV is provably secure in the strongest model — DAE — where the adversary can choose plaintexts and ciphertexts but is unable to distinguish the pair of encryption and decryption oracles from the pair of "random-bits" and "always invalid" oracles. Scheme HBS [19] and its refinement BTM [18] by Iwata and Yasuda use polynomial hash functions for MAC and a modified CTR mode. Similarly to CMAC, secondary keys are derived out of a single key by encrypting constants. HBS and BTM are provably secure in the DAE setting.

Gennaro and Halevi proposed the general template of Hash-then-Encrypt [15], which may be viewed as weakening of SIV. Here a PRF is replaced with a hash function, which might not even be collision resistant. for instance, they showed that a composition of universal hashing and CTR mode is secure. However, the performance is gained at the cost of weakening the model. Confidentiality is achieved in the assumption of random plaintexts (RPA), where the adversary obtains two plaintexts (out of his control) and one of corresponding ciphertexts, and he has to guess which one. This "left-or-right" setting is provably weaker than DAE, but is still sufficient for key wrapping schemes where inputs have enough entropy. Osaki and Iwata [22] continued work in this direction and introduced a special class of universal hash functions which are fine to use with ECB or CBC.

The Keccak team presented an authenticated encryption mode SpongeWrap, which under certain circumstances and proper formatting of plaintext and headers may serve for the key wrapping [4]. However, the paper [4] is quite vague on this topic, and it was later confirmed [1] that the key wrapping was not among the main applications of SpongeWrap. Quite recently, a misuse-resistant authenticated encryption scheme APE(X) has been presented [8], but the paper was not available to the author at the time of submission.

On Even-Mansour ciphers. Our construction may resemble a variant of the Even-Mansour cipher [13], where a single permutation $\mathcal{F}$ is turned to a blockcipher $\mathcal{E}_K(X) = K \oplus \mathcal{F}(\mathcal{X} \oplus \mathcal{K})$. The resulting cipher is provably secure up to $2^{n/2}$ queries to both blockcipher and permutation [12] when the key is as wide as the permutation. It may be tempting to construct a key wrapping scheme by taking a wide permutation and a short key, and encrypt a plaintext with some redundancy. This would reduce the security of the whole scheme to the PRP security of the Even-Mansour cipher and hence provide a desirable proof in the standard model.

However, this approach is dangerous, as reducing the key length also reduces the overall security. Indeed, when the key length $k < n$, one may ask to encrypt $2^{k/2}$ plaintexts which are constant in the bytes not touched by the key. In the offline stage, an adversary also applies the permutation to these inputs and searches for a collision between two groups of outputs in the last $n-k$ bits. This allows to recover the key with complexity $2^{k/2}$ whereas our construction does not have a security loss up to 2^k operations. Therefore, an Even-Mansour cipher can not be used as is, and a more sophisticated PRP candidate would be needed to get a scheme provably secure in the standard model.

Outline of the paper. We recall the syntax of key wrapping schemes and relevant security definitions in Section 2. Then we describe our proposal KWF in Section 3 and also recommend a set of permutations for various security levels. We immediately proceed with the security proof of KWF in Section 4. We survey the existing key wrapping schemes in the Conclusion.

2 Syntax and security of key wrapping schemes

2.1 Syntax

A *key wrapping scheme* Π is a symmetric authenticated encryption scheme and is defined as a pair of functions $\mathcal{E}$ and $\mathcal{D}$, which provide encryption and decryption, respectively. The secret key K, shared between parties, belongs to the key space $\mathcal{K}$. The encryption function takes the key to be wrapped from the message space $\mathcal{M}$ and encrypts it to a ciphertext $C \in \mathcal{C}$. If the scheme is able to authenticate some data without encrypting it, it is called the associated data (AD). The associated data space is denoted by $\mathcal{H}$, hence

$$\mathcal{E}: \mathcal{K} \times \mathcal{H} \times \mathcal{M} \rightarrow \mathcal{C}.$$

The decryption function takes a key, a ciphertext, and possibly associated data as input and returns either a plaintext (wrapped key) from the message space or the invalid message $\perp$:

$$\mathcal{D}: \mathcal{K} \times \mathcal{H} \times \mathcal{C} \rightarrow \mathcal{M} \cup \{\perp\}.$$

The key wrapping scheme differs from the probabilistic (regular) authenticated encryption schemes as it does not use any random numbers or nonces.

Clearly, the produced ciphertext decrypts back to the plaintext if the same associated data is used:

$$\mathcal{D}\,(K, H, (\mathcal{E}(K, H, M))) = M.$$

The purpose of a key wrapping scheme, as well as of a regular authenticated encryption scheme, is to achieve:

– Privacy by making all the ciphertexts "look randomly".
– Data authenticity by making all the ciphertexts not produced by the key owner "decrypt to invalid".

Hence the receiver may verify that the unwrapped key is authentic as otherwise the ciphertext would decrypt to invalid. Let us introduce these notions formally.

2.2 Security

Rogaway and Shrimpton [25] were the first who introduced a single notion that amalgamates both privacy and authenticity properties of a key wrapping scheme. They called it Deterministic Authenticated Encryption (DAE). Gennaro and Halevi proposed a weaker notion of security (RPA+INT), where the plaintexts are randomly chosen and are out of adversary's control. We prove the security of KWF in a strengthened version of DAE, so our scheme is secure for general-purpose encryption where the adversary may control the plaintexts.

The security of a DAE scheme is defined as the inability to distinguish between the two worlds, where an adversary has access to two oracles [25]. One world consists of the encryption oracle $\mathcal{E}(\cdot, \cdot)$ and decryption oracle $\mathcal{D}(\cdot, \cdot)$, where the secret key is randomly chosen. The second world consists of the "random-bits" oracle $\$(\cdot, \cdot)$ and the "always-invalid" oracle $\perp(\cdot, \cdot)$ (Figure 1).

This setting serves well for encryption schemes based on secure PRPs and authentication schemes based on secure PRFs. We only have to make a couple of refinements. As we work with a single permutation, we have to additionally allow the adversary to access it. Moreover, we have to model it as randomly drawn [3, 6], which in turn requires us to assume that the permutation has no nontrivial properties.

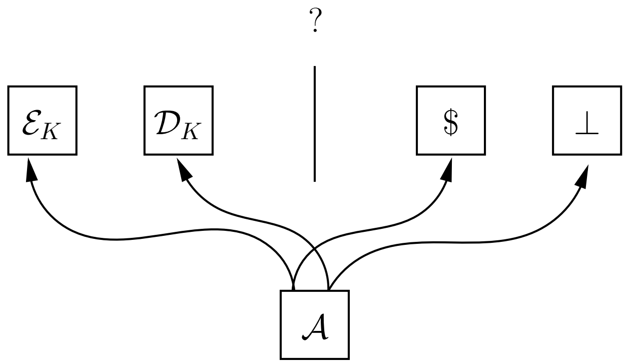

Fig. 1. Indistinguishability setting for DAE.

We also slightly refine the definition of the "random-bit" oracle $\$(\cdot,\cdot)$ with the following motivation. Since the encryption is invertible, an ideal encryption scheme with a fixed key and associated data should be a permutation. Hence it is natural to model the oracle $\$(\cdot,\cdot)$ as an ideal cipher — a set of randomly chosen permutations indexed by a key. Here the associated data serves as a key. This model allows for an increased security level and a tighter bound, since a traditional proof of security for an encryption mode invokes a PRF and then applies the PRF-to-PRP switching lemma. This lemma limits the security level with the birthday bound, which we would like to avoid.

Definition 1. *Let $\Pi = (\mathcal{K}, \mathcal{E}[\mathcal{F}], \mathcal{D}[\mathcal{F}])$ be a DAE scheme based on permutation $\mathcal{F}$. Let the adversary A have the access to $\mathcal{F}$. The* DAE advantage *of A in breaking Π is computed as follows:*

$$\mathbf{Adv}_{\Pi}^{\mathrm{dae}}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K},\, \mathcal{F}(\cdot) \xleftarrow{\$} \mathrm{Perm}(n):\ A^{\mathcal{E}_K[\mathcal{F}](\cdot,\cdot),\mathcal{D}_K[\mathcal{F}](\cdot,\cdot)} \Rightarrow 1\right] - \\ - \Pr\left[\mathcal{F}(\cdot) \xleftarrow{\$} \mathrm{Perm}(n):\ A^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right].$$

On query (H, X) the oracle $\$(\cdot,\cdot)$ returns a random string of length n so that it is a permutation (bijective function) for every H. The set of all permutations over $\{0,1\}^n$ is denoted by $\mathrm{Perm}(n)$. The $\perp(\cdot,\cdot)$ oracle always returns $\perp$ (invalid). We exclude trivial wins: the adversary shall not ask (H, Y) of its right oracle if some previous left oracle query of (H, X) returned Y and vice versa. Without loss of generality, the adversary does not repeat a query and does not ask left queries outside of $\mathcal{H} \times \mathcal{M}$. Here and in the further text we implicitly assume that $\mathcal{F}^{-1}$ is available together with $\mathcal{F}$.

The maximum advantage as a function of the number of allowed queries is the natural quantitative measure of the security of a key wrapping schemes:

$$\mathbf{Adv}_{\Pi}^{\mathrm{dae}}(q) \stackrel{\mathrm{def}}{=} \max_A \mathbf{Adv}_{\Pi}^{\mathrm{dae}}(A),$$

where we take maximum over all adversaries asking at maximum q queries to all oracles.

3 Our proposal: KWF

3.1 Notation

For two bit strings X and Y of the same length, $X \oplus Y$ is their xor. For an integer $n \geq 1$, $\{0,1\}^n$ is the set of all bit strings of n bits, and $\{0,1\}^{m..n}$ is the set of strings of m to n bits long. Also $X_{m..n}$ denotes the substring of X containing bits with indices from m to n, where the first index is 1. We write $X \stackrel{\$}{\leftarrow} \mathcal{X}$ for sampling an element from the set $\mathcal{X}$ uniformly at random.

3.2 Description of KWF

Our scheme provides an authenticated encryption of short messages and is based on a fixed n-bit permutation $\mathcal{F}$. Of the n-bit input, k bits are devoted to the key K, l bits to the associated data H, and $n-k-l$ to the message M. As the associated data needs only authentication, we map a possibly long string H to an l-bit value with a cryptographic hash function $\mathcal{G}$. $\mathcal{G}$ should be collision-resistant, and it should return some valid output on empty input (if it requires redefinition of $\mathcal{G}$, the new function shall still be collision-resistant).

We define scheme KWF formally as $\Pi = (\mathcal{K}, \mathcal{E}[\mathcal{F}], \mathcal{D}[\mathcal{F}])$, where:

1. $\mathcal{K} = \{0,1\}^k$ — the key space;
   $\mathcal{H} = \{0,1\}^{0..t}$ — the associated data (AD) space;
   $\mathcal{M} = \{0,1\}^{1..(n-k-l-1)}$ — the message space;
   $\mathcal{C} = \{0,1\}^n$ — the ciphertext space.
2. $\mathcal{G} : \{0,1\}^{0..t} \longrightarrow \{0,1\}^l$ — hash function for the associated data;
   pad : $\{0,1\}^{1..(n-k-l-1)} \longrightarrow \{0,1\}^{n-k-l}$ — invertible padding function;
   $\mathcal{F} : \{0,1\}^n \longrightarrow \{0,1\}^n$ — fixed permutation.
3. $\mathcal{E}[\mathcal{F}] : \mathcal{K} \times \mathcal{H} \times \mathcal{M} \longrightarrow \mathcal{C}$ — encryption function (Figure 2):

$$\Big\{ \mathcal{E}_K[\mathcal{F}](H, M) = \mathcal{F}\left(K || \mathcal{G}(H) || pad[M]\right)\right) \oplus \left(K || 0^{n-k}\right).$$

4. $\mathcal{D}[\mathcal{F}] : \mathcal{K} \times \mathcal{H} \times \{0,1\}^n \longrightarrow \mathcal{M} \cup \{\perp\}$ — decryption function. The output $\mathcal{D}_K[\mathcal{F}](H, C)$ is computed as follows:
   (a) $X \leftarrow \mathcal{F}^{-1}(C \oplus \left(K || 0^{n-k}\right))$.
   (b) If $X_{1..k} \neq K$, return $\perp$.
   (c) If $X_{k+1..k+l} \neq \mathcal{G}(H)$ return $\perp$.
   (d) Return $\mathrm{pad}^{-1}(X_{k+l+1..n})$.

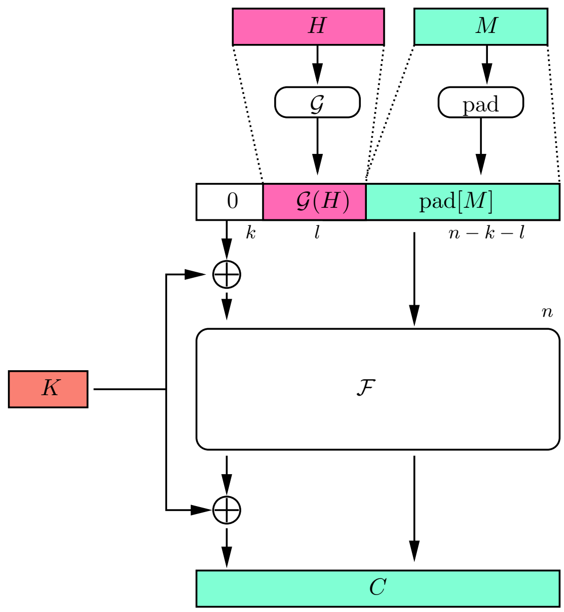

Fig. 2. Our proposal: KWF.

3.3 Recommended parameters

In Section 4 we prove that the adversary asking at most q encryption and decryption queries has the maximum advantage of

$$2\mathbf{Adv}_{\mathcal{G}}^{\mathrm{coll}}(q) + \frac{2.5q}{2^k} + \frac{8.5q^2}{2^{n-k}},$$

where $\mathbf{Adv}_{\mathcal{G}}^{\mathrm{coll}}(q)$ is the maximum advantage of the adversary trying to violate the collision resistance of $\mathcal{G}$ after making q queries to $\mathcal{G}$ (a more rigorous reduction to collision resistance is presented in Section B. We assume that for a collision-resistant hash function:

$$\mathbf{Adv}_{\mathcal{G}}^{\mathrm{coll}}(q) \leq \frac{q^2}{2^l}.$$

Then to get the security level of S bits, the following constraints are sufficient:

$$k \geq S, \quad l \geq 2S, \quad n \geq 2S + k.$$

Therefore, for 80-bit master keys and 80-bit (padded) plaintexts we need $n \geq 320$, and for 128-bit keys and plaintexts — $n \geq 512$.

There are several families of sponge hash functions that provide suitable permutations for KWF. The Keccak family [5] uses permutations of width $n = 25 \cdot 2^l$, $l = 0..6$, with $n = 1600$ chosen for SHA-3. Keccak permutations are reasonably efficient in software and hardware, and are natural choice whenever SHA-3 is implemented on the platform. We propose to use $n = 400, 800, 1600$ and denote them as Keccak-n. It is natural to use the Keccak hash function also for the associated data. By Keccak/t we denote the sponge hash function using the 1600-bit Keccak permutation with capacity $2t$ and rate $1600-2t$. The function Keccak/256 is to be standardized as SHA-3-256.

As far as we know, the key wrapping schemes are also deployed on smart cards. Hence we would like to offer a portfolio of permutations for KWF that are suitable for resource-constrained platforms. Quark [2], Photon [16], and Spongent [9] are recently proposed families of lightweight hash functions based on the sponge construction. They use permutations from 88 to 768 bits wide and were not shown any internal weaknesses. Depending on the message length and the security level, permutations from 256 to 768 bits can be recommended for KWF. Some more details are given in Appendix A and the summary in Table 2.

Whenever SHA-2 is implemented on the platform, it also can be used as the hash function for the associated data. The most suitable for KWF are SHA-224 and SHA-256.

Some permutations are significantly faster than their inverses, e.g., the Keccak permutations. Assuming that the receiver in the key wrapping scheme is more resource-constrained, we propose to use the inverse of such permutation for encryption, and hence the forward call for the decryption.

4 Security of KWF

Our main result states that if an adversary can not violate the collision resistance of G and makes fewer than $\min(2^{(n-k)/2}, 2^k)$ queries, she is unlikely to violate the security of KWF as a DAE scheme. The term $\mathbf{Adv}_{\mathcal{G}}^{\text{coll}}(q)$ in our bound quantifies the ability of the adversary making at most q queries to $\mathcal{G}$ to find a colliding pair. A more rigorous formulation of our results can be found in Appendix.

Theorem 1. *The DAE advantage of an adversary attacking KWF and asking the total of at most q queries to all oracles and $\mathcal{F}$ is bounded as follows:*

$$\mathbf{Adv}_{\Pi}^{\text{dae}}(q) \leq 2\mathbf{Adv}_{\mathcal{G}}^{\text{coll}}(q) + \frac{2.5q}{2^k} + \frac{8.5q^2}{2^{n-k}}.$$

Proof. We split this expression in two following the approach in [25]:

$$\Pr\left[K \xleftarrow{\$} \mathcal{K},\, \mathcal{F}(\cdot) \xleftarrow{\$} \mathrm{Perm}(n):\ A^{\mathcal{E}_K[\mathcal{F}],\mathcal{D}_K[\mathcal{F}]} \Rightarrow 1\right] - $$
$$- \Pr\left[\mathcal{F}(\cdot) \xleftarrow{\$} \mathrm{Perm}(n):\ A^{\$,\perp} \Rightarrow 1\right] =$$
$$= \underbrace{\Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\mathcal{D}_K[\mathcal{F}]} \Rightarrow 1\right] - \Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\perp} \Rightarrow 1\right]}_{p_1} +$$
$$+ \underbrace{\Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\perp} \Rightarrow 1\right] - \Pr\left[A^{\$,\perp} \Rightarrow 1\right]}_{p_2}.$$

4.1 Bounding p_1 (authenticity proof)

Consider

$$p_1 = \Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\mathcal{D}_K[\mathcal{F}]} \Rightarrow 1\right] - \Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\perp} \Rightarrow 1\right],$$

where K and $\mathcal{F}$ are randomly chosen.

We assume without loss of generality that A halts and outputs 1 whenever the right oracle returns $M \neq \perp$. Prior to this event, both oracle pairs behave identically as $(\mathcal{E}_K[\mathcal{F}], \perp)$. Therefore, p_1 is bounded by the probability that A asks a right-oracle query (H, Y) so that $\mathcal{D}_K(H, Y) \neq \perp$.

Let us denote the set of ciphertexts obtained prior to this query by $\overline{\mathcal{C}}$, the set of $\mathcal{F}$ responses and $\mathcal{F}^{-1}$ queries by $\mathcal{F}_o$, and of $\mathcal{F}$ queries and $\mathcal{F}^{-1}$ responses by $\mathcal{F}_i$.

By definition, the adversary is unable to use a pair (H, Y) where Y has been a response $Y = \mathcal{E}_K(H, X)$ for some X. Hence either $Y \notin \overline{\mathcal{C}}$, or $Y = \mathcal{E}_K(H', X), H' \neq H$. In the latter case the ciphertext decrypts to $\mathcal{G}(H')||X$, so the decryption returns $\perp$ unless H and H' form a collision pair for $\mathcal{G}$. Here the success rate of the collision search for $\mathcal{G}$ is bounded by $\mathbf{Adv}_{\mathcal{G}}^{\mathrm{coll}}(q)$.

If $Y \notin \overline{\mathcal{C}}$, then

$$\begin{aligned}\Pr[\mathcal{D}_K(H,Y) \neq \perp] \leq\ & \Pr[\mathcal{D}_K(H,Y) \neq \perp \mid (Y \oplus K) \notin \mathcal{F}_o] + \\ & + \Pr[\mathcal{D}_K(H,Y) \neq \perp \mid (Y \oplus K) \in \mathcal{F}_o].\end{aligned}$$

Let us now estimate both addends of the right side.

- $(Y \oplus K) \notin \mathcal{F}_o$ (here and further we shortly write $Y \oplus K$ instead of $Y \oplus (K||0^{n-k})$). Then the permutation $\mathcal{F}^{-1}$ is asked with a fresh query, so its output is uniformly distributed along previously unallocated values. The decryption returns invalid if

  $$F^{-1}(Y \oplus K)_{1..k+l} \neq K||G(H).$$

  Hence at maximum 2^{n-k-l} values pass this condition. As at minimum $2^n - q$ values remain unassigned, we obtain

  $$\Pr[\mathcal{D}_K(H,Y) \neq \perp \mid (Y \oplus K) \notin \mathcal{F}_o] \leq \frac{2^{n-k-l}}{2^n - q}. \tag{1}$$

- $(Y \oplus K) \in \mathcal{F}_o$. Hence the decryption oracle ask the permutation with a query that is not fresh. Then the decryption returns $\perp$ if

  $$\mathcal{F}^{-1}(Y \oplus K) \neq K||Z$$

  for any Z. We say that the *input clash* (IC) occurs, if $(K||Z) \in \mathcal{F}_i$ for some Z. Hence without the input clash the decryption error is guaranteed:

  $$\Pr[\mathcal{D}_K(H,Y) \neq \perp \mid (Y \oplus K) \in \mathcal{F}_o,\ \text{no IC occurred}] = 0.$$

Finally,

$$\Pr\left[\mathcal{D}_K(H,Y) \neq \perp \mid \text{no IC occurred}\right] \leq \frac{2^{n-k-l}}{2^n - q}.$$

and

$$p_1 = \mathbf{Adv}_{\mathcal{G}}^{\mathrm{coll}}(q) + \left[1 - \left(1 - \frac{2^{n-k-l}}{2^n - q}\right)^q\right] + \Pr(IC; q) \leq$$
$$\leq \mathbf{Adv}_{\mathcal{G}}^{\mathrm{coll}}(q) + \frac{2q}{2^{k+l}} + \Pr(IC; q).$$

It remains to bound the probability $\Pr(IC; q)$ of getting the input clash after q queries. Here either the adversary tries to guess the key in $\mathcal{F}$ queries or hopes to obtain it as a prefix in $\mathcal{F}^{-1}$ responses. In the worst case, when all the q prefixes are different, the probability of having K among them is bounded by $q/2^k$, so we have the following bound on the input clash:

$$\Pr(IC; q) \leq \frac{q}{2^k}. \tag{2}$$

This gives the final bound on p_1:

$$p_1 \leq \mathbf{Adv}_{\mathcal{G}}^{\mathrm{coll}}(q) + \frac{2q}{2^{k+l}} + \frac{q}{2^k}. \tag{3}$$

4.2 Bounding p_2 (privacy proof)

Recall that

$$p_2 = \Pr\left[A^{\mathcal{E}_K[\mathcal{F}],\perp} \Rightarrow 1\right] - \Pr\left[A^{\$,\perp} \Rightarrow 1\right],$$

where K and $\mathcal{F}$ are chosen randomly.

We can drop the oracle $\perp$ simply by considering the adversary B that has access to the left oracle only and runs A. She transfers queries of A directly to the oracles and returns $\perp$ to all queries by A to the right oracle. Hence

$$p_2 \leq \Pr\left[B^{\mathcal{E}_K[\mathcal{F}]} \Rightarrow 1\right] - \Pr\left[B^{\$} \Rightarrow 1\right].$$

In the further text we show that in the absence of two events, so called the output clash and the oracle repetition, oracles $\mathcal{E}_K[\mathcal{F}]$ and \$ produce identically distributed results and hence are indistinguishable. The clashes and some remarks on how they can be exploited are depicted in Figure 3.

Let us go to the details. Consider the query (H, X) to the encryption oracle $\mathcal{E}$ or \$. Denote by $\overline{\mathcal{C}}$ the set of encryption oracle responses obtained beforehand. We are also interested in the last $(n-k)$ bits of ciphertext, which are unaffected by the key addition. We denote $C|_{k+1..n}$ by $\widehat{C}$ and use the same notation for $(n-k)$-bit suffixes of $\mathcal{F}_o$ denoted by $\widehat{\mathcal{F}_o}$.

Let us say that the encryption oracle response $C = \mathcal{E}_K(H, X)$ causes the *output clash* if $\widehat{C} \in \widehat{\mathcal{F}_o}$, i.e. the ciphertext collides with one of stored permutation

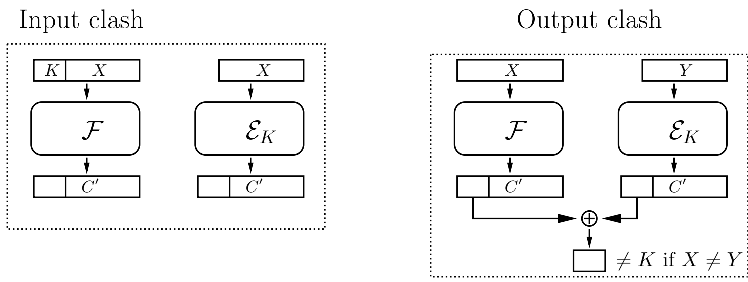

Fig. 3. Detection and exploit of the clashes. In case of input clash we encrypt $(n-k)$-bit suffix X of the permutation query $K||X$ and detect suffix collision in outputs. In case of output clash some ciphertext prefixes are forbidden, which allow to distinguish the encryption from the ideal cipher.

outputs on the last $n-k$ bits. Let $\mathcal{V}$ be the set of the ciphertexts that did not occur in previous responses and do not cause the output clash:

$$\mathcal{V} \stackrel{\text{def}}{=} \left[\{0,1\}^k \times \left(\{0,1\}^{n-k} \setminus \widehat{\mathcal{F}_o}\right)\right] \setminus \overline{E}.$$

Lemma 1. *If the input clash did not occur and $\mathcal{G}(H)$ does not collide with any previous such value, then all responses $\mathcal{E}_K(H,X) \in \mathcal{V}$ are equiprobable.*

Proof. Since there was no input clash and no collision, the input $K||\mathcal{G}(H)||pad[X]$ has not been queried to the permutation. Hence, it is fresh, and its output is uniformly taken from $\{0,1\}^n \setminus \overline{\mathcal{C}}$. It remains to remove from this set the values that cause the output clash, i.e. the set $\{0,1\}^k \times \left(\{0,1\}^{n-k} \setminus \widehat{\mathcal{F}_o}\right)$. This concludes the proof.

The ideal cipher has a different distribution, as its outputs may collide for distinct H. Let us say that the *oracle repetition* (OR) occurs if $\$(H,X) = \$(H',X')$ for some previously queried (H',X'). If we exclude the oracle repetition and output clash events, the distribution will be the same:

$$\$(H,X) \xleftarrow{\$\,|\,\text{no OR, no OC, no Coll}} \mathcal{V}.$$

Combining with Lemma 1, we conclude that if no input clash, no output clash, and no oracle repetition occurs, the encryption oracles produce identically distributed responses and are indistinguishable. Therefore,

$$p_2 \leq \Pr(IC;q) + \Pr(\text{OC for } \mathcal{E}\,|\,\text{no IC}; q) + \Pr(\text{OC for } \$(\cdot,\cdot); q) + \\ + \Pr(OR) + \mathbf{Adv}^{\text{coll}}_{\mathcal{G}}(q). \quad (4)$$

Output clash bound. Provided no input clash and no collision in $\mathcal{G}$, a query $\mathcal{E}(H, X)$ yields a fresh query to $\mathcal{F}$. Hence the output is uniformly distributed among at least $2^n - q$ previously unassigned values, of which at maximum $q \times 2^k$ cause the output clash. Therefore, assuming $q \leq 2^k$

$$\Pr\left(\mathcal{E}(\widehat{H, X}) \in \widehat{\mathcal{F}_o}\right) \leq \frac{q 2^k}{2^n - q} \leq \frac{2q}{2^{n-k}}.$$

Thus we have the following bound:

$$\Pr(\text{OC for } \mathcal{E} \mid \text{no IC}; q) \leq 1 - \left(1 - \frac{2q}{2^{n-k}}\right)^q \leq \frac{4q^2}{2^{n-k}}. \tag{5}$$

The bound for the ideal cipher is the same:

$$\Pr\left(\$(\widehat{H, X}) \in \widehat{\mathcal{F}_o}\right) \leq \frac{q \cdot 2^k}{2^n - q} \implies \Pr(\text{OC for } \$(\cdot,\cdot); q) \leq \frac{4q^2}{2^{n-k}}. \tag{6}$$

Oracle repetition bound. We calculate the probability of the event that during q queries to the PRP-oracle \$ there is no collision in outputs. Consider i-th query. The oracle chooses its output uniformly out of at least $(2^n - q)$ possibilities, of which at maximum q cause a collision. Hence

$$\Pr(OR; q) = 1 - \prod_{i=1}^{q} \left(\frac{2^n - 2q}{2^n - q}\right) = 1 - \left(1 - \frac{q}{2^n - q}\right)^q \leq 1 - e^{-\frac{2q^2}{2^n}} \leq \frac{3q^2}{2^n}. \tag{7}$$

We substitute Equations (2), (5), (6), and (7) to Equation (8) and obtain the final bound

$$p_2 \leq \frac{q}{2^k} + \frac{8q^2}{2^{n-k}} + \frac{3q^2}{2^n} + \mathbf{Adv}_{\mathcal{G}}^{\text{coll}}(q). \tag{8}$$

Then we sum Equations (3) and (8) and get:

$$\mathbf{Adv}_{\Pi}^{\text{dae}}(q) \leq \mathbf{Adv}_{\mathcal{G}}^{\text{coll}}(q) + \frac{2q}{2^{k+l}} + \frac{q}{2^k} + \frac{q}{2^k} + \frac{8q^2}{2^{n-k}} + \frac{3q^2}{2^n} + \mathbf{Adv}_{\mathcal{G}}^{\text{coll}}(q) \leq$$
$$\leq 2\mathbf{Adv}_{\mathcal{G}}^{\text{coll}}(q) + \frac{2.5q}{2^k} + \frac{8.5q^2}{2^{n-k}}.$$

This concludes the proof of Theorem 1.

5 Acknowledgements

The author thanks Alex Biryukov, Jean-Sébastien Coron, Joan Daemen, and Gilles Van Assche for valuable comments on the paper.

6 Conclusion

We have described a new key wrapping scheme — KWF. It is based on a single fixed permutation, with a lot of candidate permutations available in sponge designs: Keccak/SHA-3, Quark, Spongent, Photon. Though keys to be wrapped are limited to the length of 1411 bits, our scheme still provides a simple and efficient key update protocol for most of symmetric and several asymmetric cryptosystems. It can also bind the associated data to the message, and is able to preprocess it without the master key. The ciphertext length is equal to the permutation width n, while the master key length k can vary.

We recalled several alternative schemes and concluded that KWF is a viable alternative when a user wants to achieve 128-bit security within a simple design. Though the security of KWF is proven in the random-permutation model, we showed that no similar schemes secure in the standard model have the same features, and the Even-Mansour construction needs a careful amplification to suit the 128-bit security requirement.

The scheme is provably secure in the refined concept of DAE, where we add the adversarial access to the permutation. Assuming no weakness in the permutation and a collision resistant hash function for the associated data, the violation of DAE property is unlikely for the number of queries $q \leq 2^k$. Hence the security level of 128 bits is easy to deliver, which is not the case for other key wrapping schemes using AES.

A scalable version of our scheme, which would process arbitrarily long messages and remain simple and secure, is an object for the future work.

We believe that there are numerous different applications where DAE schemes can be used. Whereas one might need 128-bit security level at first, another would want to process arbitrary long messages. We have constructed a comparative table to demonstrate that KWF may find its own niche (Table 1). The overhead is here defined as the ratio of extra blockcipher calls compared to the sole encryption of the same data; we simply divide the permutation width by the message length. The expansion is the the number of extra bits compared to the ciphertext of the same data. We also compare the need of block cipher or a hash function, the ability to preprocess the header in advance, and the ability to deliver 128-bit security with recommended parameters (usually AES for other schemes).

References

1. Gilles Van Assche. Private communication, August 2013.
2. Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A lightweight hash. In *CHES'10*, volume 6225 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2010. `https://131002.net/quark/quark_full.pdf`.
3. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *EUROCRYPT'08*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

	KWF	HBS	AES-KW	SIV	Hash-then-E
Message length	0..1411	Arbitrary			
Overhead	1.2–1.5	2	12	2	1+ε
Expansion	192–384	128	$\|H\| + 64$	128	128
Security model	DAE	DAE	No	DAE	RPA
Security proof	RP	PRP,PRF	No	PRP, PRF	PRP
Block cipher	No	Yes	Yes	Yes	Yes
Hash function	Yes	Maybe	No	Yes	Yes
Preprocess header	Yes	No	No	Yes	Yes
128-bit security	Yes	Not with AES	No	Not with AES	

Table 1. KWF and other key wrapping schemes

4. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *Selected Areas in Cryptography'11*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.
5. Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. The Keccak reference, version 3.0, 2011. `http://keccak.noekeon.org/Keccak-reference-3.0.pdf`.
6. Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. Security analysis of the mode of JH hash function. In *FSE'10*, volume 6147 of *Lecture Notes in Computer Science*, pages 168–191. Springer, 2010.
7. Alex Biryukov and Dmitry Khovratovich. PPAE: Parallelizable permutation-based authenticated encryption, 2013. presented at DIAC 2013: `http://2013.diac.cr.yp.to/slides/khovratovich.pdf`.
8. Andrey Bogdanov, Elena Andreeva, Bart Mennink, Nicky Mouha, Atil Luykx, and Ken Yasuda. APE(X): Authenticated permutation-based encryption with extended misuse resistance, 2013. presented at DIAC 2013: `http://2013.diac.cr.yp.to/slides/luykx.pdf`.
9. Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. Spongent: A lightweight hash function. In *CHES'11*, volume 6917 of *Lecture Notes in Computer Science*, pages 312–325. Springer, 2011.
10. Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert, John P. Steinberger, and Elmar Tischhauser. Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations. In *EUROCRYPT'12*, volume 7237 of *Lecture Notes in Computer Science*, pages 45–62. Springer, 2012.
11. Joan Daemen. Permutation-based symmetric cryptography and Keccak. Technical report, Ecrypt II, Crypto for 2020 Invited Talk, 2013. `https://www.cosic.esat.kuleuven.be/ecrypt/cryptofor2020`.
12. Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In *EUROCRYPT'12*, volume 7237 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2012.

13. Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. In *ASIACRYPT'91*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 1991.
14. Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein hash function family, `http://www.skein-hash.info/sites/default/files/skein1.3.pdf`. Submission to NIST (Round 3), 2010.
15. Rosario Gennaro and Shai Halevi. More on key wrapping. In *SAC'09*, volume 5867 of *Lecture Notes in Computer Science*, pages 53–70. Springer, 2009.
16. Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In *CRYPTO'11*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011. `https://sites.google.com/site/photonhashfunction`.
17. Tetsu Iwata, Keisuke Ohashi, and Kazuhiko Minematsu. Breaking and repairing GCM security proofs. In *CRYPTO'12*, volume 7417 of *Lecture Notes in Computer Science*, pages 31–49. Springer, 2012.
18. Tetsu Iwata and Kan Yasuda. BTM: A single-key, inverse-cipher-free mode for deterministic authenticated encryption. In *SAC'09*, volume 5867 of *Lecture Notes in Computer Science*, pages 313–330. Springer, 2009.
19. Tetsu Iwata and Kan Yasuda. HBS: A single-key mode of operation for deterministic authenticated encryption. In *FSE'09*, volume 5665 of *Lecture Notes in Computer Science*, pages 394–415. Springer, 2009.
20. Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In *FSE'11*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.
21. NIST. Special publication 800-38f: Recommendation for block cipher modes of operation: Methods for key wrapping, 2008. `http://csrc.nist.gov/publications/drafts/800-38F`.
22. Yasushi Osaki and Tetsu Iwata. Further more on key wrapping. *IEICE Transactions*, 95-A(1):8–20, 2012. Also published at SKEW 2011: `http://skew2011.mat.dtu.dk`.
23. Phillip Rogaway. Authenticated-encryption with associated-data. In *ACM Conference on Computer and Communications Security'02*, pages 98–107, 2002.
24. Phillip Rogaway. Formalizing human ignorance. In *VIETCRYPT'06*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228. Springer, 2006.
25. Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In *EUROCRYPT'06*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.

A Lightweight permutations for KWF

Here we recommend for KWF some particular permutations taken from the lightweight hash function families: Quark, Photon, Spongent.

The Quark family [2] offers compact and low-power hash functions best suitable for RFID technology and other resource-constrained platforms. It uses permutations of width $n = 136, 176, 256$. Here $n = 256$ (s-Quark) is a good candidate for KWF operating on 64-bit keys and plaintexts, and u- and d-Quark ($n = 136, 176$) can be chosen as the hash function $\mathcal{G}$ with the 64-bit and 80-bit level of collision resistance, respectively. We denote Quark permutations of width n by Quark-n.

The Photon family [16] is an AES-based lightweight hash function design with permutations of width 100, 144, 196, 256, 288. We denote its permutations of width n by Photon-n and recommend using Photon-288 for applications with 64-bit level security. The hash functions Photon/n offer $n/2$-bit of security for collision resistance and are suitable for processing associated data.

The Spongent family [9] is another lightweight hash function design with permutations of width from 88 to 768 bits, with all widths multiple of 4 bits available. Its permutations Spongent-n can be used for all security levels, with a particular choice optimized for the input length.

<table>
<tr><th>Hash function for AD</th><th>Message length</th><th>Permutation</th></tr>
<tr><td colspan="3">Security level and key length: 64</td></tr>
<tr><td>Keccak/128, u-Quark, Photon/128</td><td>0..63</td><td>Keccak-400, Quark-256,
Photon-288, Spongent-256</td></tr>
<tr><td>Keccak/128, u-Quark, Photon/128</td><td>64..212</td><td>Keccak-400, Spongent-400</td></tr>
<tr><td>Keccak/128, u-Quark, Photon/128</td><td>213..612</td><td>Keccak-800</td></tr>
<tr><td>Keccak/128, u-Quark, Photon/128</td><td>613..1412</td><td>Keccak-1600</td></tr>
<tr><td colspan="3">Security level and key length: 80</td></tr>
<tr><td>Keccak/160, d-Quark, Photon/160</td><td>0..160</td><td>Keccak-400, Spongent-400</td></tr>
<tr><td>Keccak/160, d-Quark, Photon/160</td><td>161..560</td><td>Keccak-800</td></tr>
<tr><td>Keccak/160, d-Quark, Photon/160</td><td>561..1360</td><td>Keccak-1600</td></tr>
<tr><td colspan="3">Security level and key length: 112</td></tr>
<tr><td>SHA-224, Keccak/224, s-Quark</td><td>0..464</td><td>Keccak-800</td></tr>
<tr><td>SHA-224, Keccak/224, s-Quark</td><td>465..1264</td><td>Keccak-1600</td></tr>
<tr><td colspan="3">Security level and key length: 128</td></tr>
<tr><td>SHA-256, Keccak/256</td><td>0..415</td><td>Keccak-800</td></tr>
<tr><td>SHA-256, Keccak/256</td><td>416..1215</td><td>Keccak-1600</td></tr>
</table>

Table 2. Recommended parameters for scheme KWF. All the numbers are bit lengths.

B Security of KWF via the concept of human ignorance

Now we attempt to deal more rigorously with the collision resistance of the hash function $\mathcal{G}$ that processes the header.

We can do this by following the "human ignorance" concept introduced by Rogaway [24]. First, we introduce the variant of our scheme called KWF'. Then we show that for any adversary A breaking KWF we can explicitly construct an adversary C violating the collision resistance of $\mathcal{G}$ and an adversary B violating the DAE security of KWF' so that their total advantage exceeds the advantage of A. Then we bound the advantage of B.

Let KWF' be the version of KWF, where the headers are l-bit long and the function G is an identity function. Hence $\mathcal{G}(H)$ can be simply replaced by H.

Lemma 2. *The DAE advantage of an adversary attacking instantiation Π' of KWF' and asking the total of at most q queries to all oracles and $\mathcal{F}$ is bounded as follows:*

$$\boldsymbol{Adv}^{dae}_{\Pi'}(q) \leq \frac{2.5q}{2^k} + \frac{8.5q^2}{2^{n-k}}.$$

Proof. The proof of this lemma repeats the proof of Theorem 1 with a small refinement: $\mathcal{G}(H)$ is everywhere replaced with H, so distinct H always yield distinct inputs to $\mathcal{E}$ and $\mathcal{F}$. Hence the overall bound remains the same with the collision search term removed.

Lemma 3. *There exist (and are explicitly constructed in the proof) the adversary B attacking Π' and the adversary C attacking the collision resistance of a hash function such that for any $\mathcal{G}$ and any adversary A attacking instantiation Π of KWF*

$$\boldsymbol{Adv}^{dae}_{\Pi}(A) \leq \boldsymbol{Adv}^{dae}_{\Pi'}(B^{A,\mathcal{G}}) + \boldsymbol{Adv}^{coll}_{\mathcal{G}}(C^{A,\mathcal{G}}).$$

Adversary B asks the same number of queries as A to his oracles, and adversary C asks at maximum the same number of queries to $\mathcal{G}$ as A asks to his oracles.

Proof. The proof is very similar to the security proof of the hash-then-PRF folklore algorithm, which extends the domain of a PRF by hashing the input, in [24]. Adversary B is constructed as follows. He has access to A and $\mathcal{G}$, runs A as an oracle, and forwards all his requests to oracles $\mathcal{E}$ and $\mathcal{D}$ to his own instantiations of them for KWF'. Whenever A halts, B outputs the same bit.

Adversary C also runs A as an oracle and emulates the oracle pair $(\$, \perp)$ for him. In addition he computes and stores $\mathcal{G}(H)$ for each H queried by A. Whenever a collision is found, C halts and outputs it.

Then we try to bound $\mathbf{Adv}^{\mathrm{dae}}_{\Pi}(A) - \mathbf{Adv}^{\mathrm{dae}}_{\Pi'}(B^{A,\mathcal{G}})$. Since B and A behave identically when their worlds consist of real encryption/decryption oracles, we obtain that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{dae}}_{\Pi}(A) - \mathbf{Adv}^{\mathrm{dae}}_{\Pi'}(B^{A,\mathcal{G}}) &= \Pr\left[B^{A,\mathcal{G}}[\$, \perp]\right] - \Pr\left[A[\$, \perp]\right] \leq \\ &\leq \Pr\left[A \text{ makes queries colliding in } \mathcal{G}\right] = \mathbf{Adv}^{\mathrm{coll}}_{\mathcal{G}}(C^{A,\mathcal{G}}),\end{aligned}$$

where the last equation follows from the definition of C and the inequality comes from the fact that A and B may behave differently only if A makes queries which contain colliding H.

The next theorem immediately follows from these two lemmas and is a more rigorous version of Theorem 1.

Theorem 2. *There exist (and are explicitly constructed in the proof) the adversary B attacking KWF' and the adversary C attacking the collision resistance of a hash function such that for any $\mathcal{G}$ and any adversary A attacking KWF*

$$\boldsymbol{Adv}^{dae}_{\Pi}(A) \leq \frac{2.5q}{2^k} + \frac{8.5q^2}{2^{n-k}} + \boldsymbol{Adv}^{coll}_{\mathcal{G}}(C).$$